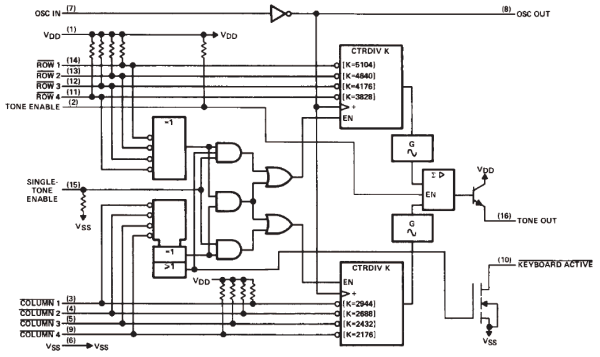
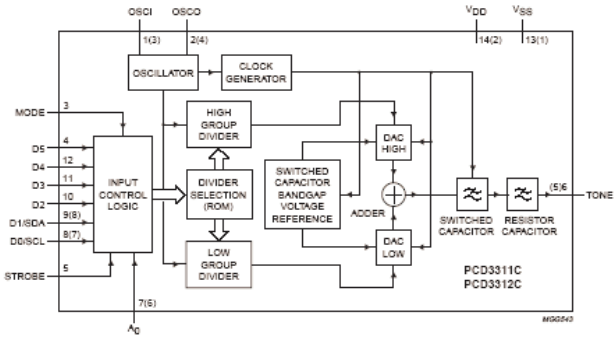
Génération par un circuit spécialisé

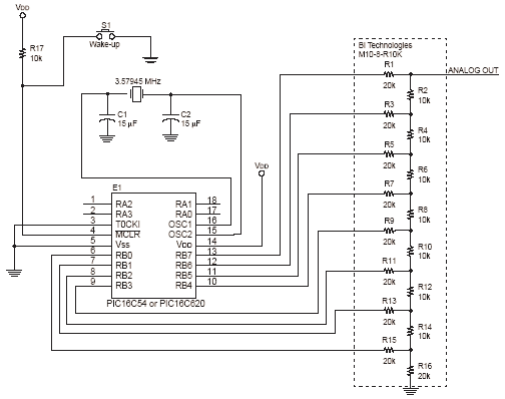
Comme par exemple le très classique TCM5089 qui réalise aussi la fonction d'encodage d'un clavier 4x4, ou un circuit plus récent, PCD3311 pilotable par I2C. Ces circuits nécessitent un quartz 3.57MHz et fournissent un niveau audio élevé, qui peut nécessiter un minimum de filtrage.





Génération par microcontrôleur et convertisseur D/A

Avec réseau R-2R ou convertisseur externe.



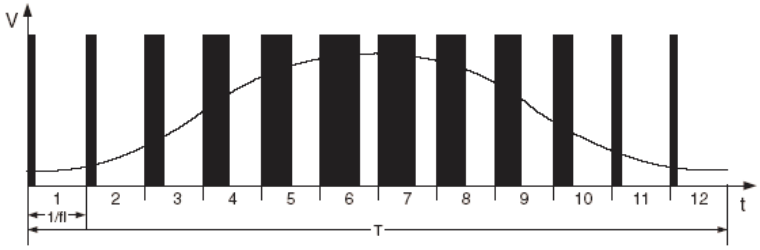
Graphique tiré de AN655 Microchip

C'est simple, ne nécessite pas un filtrage sophistiqué, mais un peu plus cher qu'un PWM simple. Mais l'inconvénient majeur est le nombre de lignes IO nécessaires.

Génération par microcontrôleur et PWM

Principe

Utiliser un Pulse Width Modulator pour générer une tension alternative est une méthode classique. A chaque échantillon, une nouvelle valeur est calculée et appliquée au PWM.



Graphique tiré de AVR314 Atmel

Générer 2 tonalités pures peut se faire avec 2 PWM indépendant ou un seul, en sommant les consignes sinus avant de l'appliquer au module PWM.

Choix de la fréquence d'échantillonnage

Traditionnellement on dit que la fréquence d'échantillonnage doit être au moins 10 fois la fréquence du signal, surtout afin que le filtre en sortie soit le plus simple possible

Résolution en fréquence

La fréquence d'échantillonnage doit être suffisamment haute pour arriver à la précision suffisante sur les fréquences de sortie. Mais celle ci dépend de la façon dont va être généré le sinus.

- Si on cherche le sinus dans une table, en incrémentant une phase entre 0 et 2π, le saut de phase entre 2 échantillons vaut $N_{tab}.F/F_{pwm}$, N_{tab} étant la longueur de la table sinus. Par exemple en considérant une table de 1024 éléments (physiquement seulement 512 échantillons seront présents car la table est symétrique) et une fréquence PWM de 39kHz, on obtient une résolution en fréquence de 38Hz, ce qui fait 5% pour la fréquence PWM la plus basse, c'est inacceptable.
On peut alors considérer une phase qui tourne sur 16 bits, et tronquer cette valeur avant d'aller chercher son sinus dans une table plus petite. Cela revient à avoir une résolution de Fpwm/65536, soit avec l'exemple ci dessus de l'ordre de 0.06 Hz, formidable.

- Si la fréquence de sortie vaut $F_{out}=F_{pwm}/N_c$, c'est à dire qu'il y a toujours un nombre entier d'échantillons dans une période du signal de sortie, alors l'erreur relative en sortie vaut $dF_{out}/F_{out}=1/N_c=F_{out}/F_{pwm}$. Donc si on veut attendre 1% de précision, il faut une fréquence d'échantillonnage 100 fois supérieure… A moins qu'on trouve un PPCM (plus petit commun multiple) à toutes les fréquences du code DTMF….

- Si le sinus est généré de façon continu (voir annexe), la précision est théoriquement parfaite, sauf qu'en pratique elle va être conditionnée par les erreurs d'arrondis qui peuvent se propager.

Résolution en amplitude

La résolution en amplitude d'un PWM dépend de la valeur maximale du compteur (PR2 dans le cas d'un PC). On a intérêt à maximaliser cette valeur, mais tous les choix ne sont pas permis car cette valeur est en général le rapport entre la fréquence d'horloge du PWM et la fréquence centrale.
Par exemple sur un PIC18 fonctionnant à 40MHz (10MHz et PLLx4), on peut considérer une fréquence d'échantillonnage à 39060 Hz pour avoir une résolution amplitude maximale sur 10 bits.

40 000 000

Fech	2 440	3 451	4 880	6 901	9 760	13 803	19 520	27 605	39 040	55 211	78 080	110 422	156 160
TMR2 prescale (1/4/16)	16	16	16	16	4	4	4	4	1	1	1	1	1
PR2	255	180	127	90	255	180	127	90	255	180	127	90	63
resolution	10,0	9,5	9,0	8,5	10,0	9,5	9,0	8,5	10,0	9,5	9,0	8,5	8,0

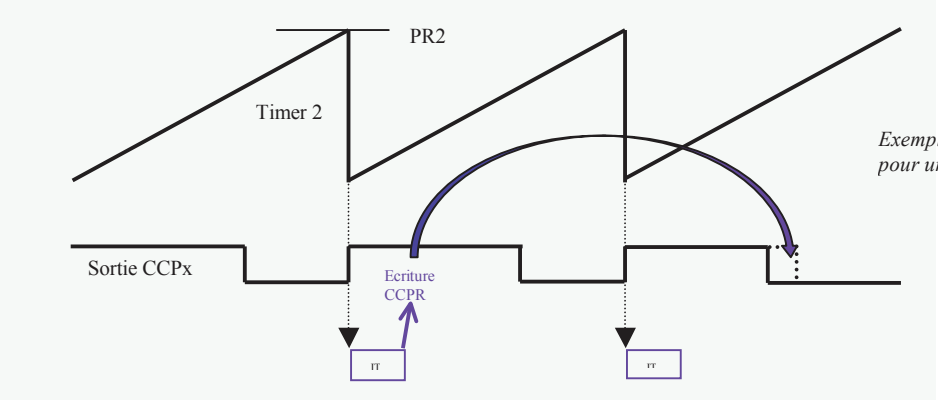
Avec une horloge à 4MHz (PIC16 standard), la situation offre moins de choix, mais on peut espérer encore 8 bits de résolution pour une fréquence d'échantillonnage proche de 16kHz.

4 000 000

Fech	2 440	3 451	4 880	6 901	9 760	13 803	19 520	27 605	39 040	55 211	78 080	110 422	156 160
TMR2 prescale (1/4/16)	4	4	1	1	1	1	1	1	1	1	1	1	1
PR2	101	71	204	144	101	71	50	35	25	17	12	8	5
resolution	8,7	8,2	9,7	9,2	8,7	8,2	7,7	7,1	6,6	6,1	5,6	5,0	4,4

Rapidité de mise à jour

Dans le cas de la génération d'un signal non continu, une contrainte sur la fréquence d'échantillonnage est présente par l'intermédiaire du délai nécessaire pour changer la consigne. A chaque échantillon, il faut répondre à l'interruption, pouvoir calculer l'échantillon suivant et mettre à jour la consigne PWM.



Exemple de timing pour un PIC16/18

Ce temps minimum peut générer un offset sur la sortie, on n'aura alors pas accès aux très faibles valeurs de rapport cyclique. Si la fréquence d'échantillonnage est trop forte, cet offset sera prohibitif et va réduire sérieusement l'amplitude en sortie.
Dans le cas d'un PIC, cette contrainte n'existe pas car la consigne PWM n'est chargée qu'à la remise à zéro du timer 2. On dispose donc de toute la période pour calculer la nouvelle consigne, il faut néanmoins que la durée de la routine d'interruption soit inférieure à la période.
Une évaluation rapide de la durée de la routine d'interruption donne une valeur proche de 100 cycles pour un PIC RISC. Si on considère que ce délai ne doit pas représenter plus de la moitié de la période PWM (et c'est déjà considérable), celle-ci ne peut être inférieure à 200 cycles, ou 800/Horloge. Cela condamne définitivement les faibles valeurs d'horloge, et plaide pour ne pas monter trop haut la fréquence PWM.

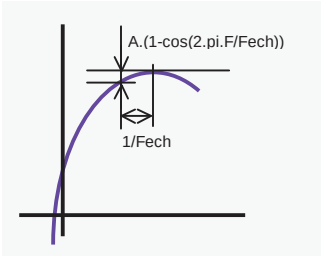
save context et call	28
phaseL=+dphaseL	2
getsinus	14
phaseH=+dphaseH	2
getsinus	14
multsign *0,75	10
add H+L	4
Update PWM	2
restore context	20
total	**96**

Sur-échantillonnage

Certains modules PWM permettent de multiplier la fréquence d'échantillonnage, en répétant N fois le même échantillons. Ainsi, la périodicité de la remise à jour de la consigne PWM est N fois plus faible que la fréquence PWM. Par exemple, sur le PIC18 on peut choisir n'importe quelle valeur entre 1 et 16, ce qui est extrêmement souple.
Evidemment cela permet de disposer de plus de temps pour calculer la nouvelle consigne, mais çà réduit la dynamique du PWM aussi. Par exemple, en considérant une fréquence PWM à 156160 Hz, et une division par 4, on retombe à 39062 Hz comme fréquence de mise à jour, mais la dynamique n'est plus de 10 bits mais seulement de 8 bits.
On voit pas trop à quoi çà sert….

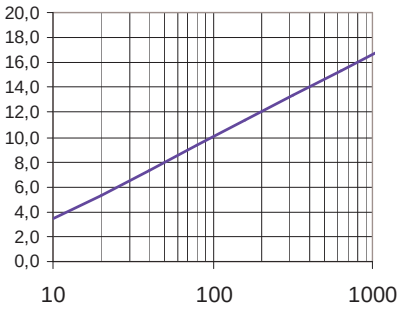
Relation amplitude-fréquence

Il ne sert à rien d'avoir un échantillonnage très précis sur l'échelle temporelle, si elle n'est pas cohérente de la résolution d'amplitude.



Le pas le plus petit en amplitude est atteint au sommet de la sinusoïde. Le nombre de niveaux en amplitude devrait être du même ordre que $1/(1-\cos(2\pi F/F_{ech}))$.

Par exemple si Fech/F vaut 50 (39062/697), le pas en amplitude vaut au minimum A/127, et donc une résolution en amplitude de 8 bits semble satisfaisante.
Si Fech/F vaut 10 (cas minimal d'un PWM), il suffit de 4 bits en amplitude…



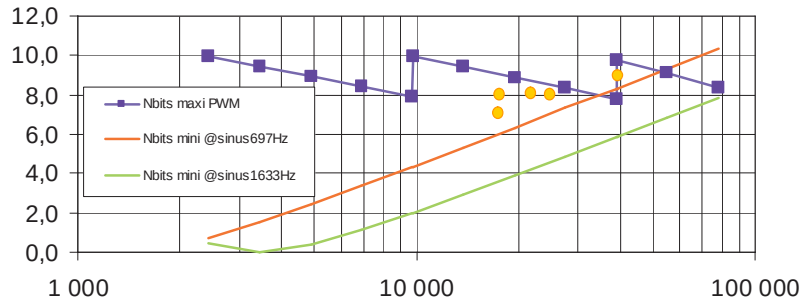
Dans le cas d'un générateur DTMF avec des fréquences entre 697 Hz et 1633 Hz, et une fréquence d'échantillonnage de 39062 Hz, une résolution en amplitude 8 bits (+/-127) est suffisante.

Choix final de la fréquence d'échantillonnage PWM

En considérant le PWM d'un PIC18F cadencé à 40MHz, le graphique ci-dessous résume les 2 contraires :

1. résolution du PWM en fonction de la fréquence d'échantillonnage, en prenant en compte la réduction due au délai de traitement de l'interruption
2. résolution minimale nécessaire pour reproduire correctement un sinus. On prendre au moins un niveau au dessous de ce minimum.

On constate que finalement on n'a pas énormément de points de fonctionnement possibles en considérant une fréquence PWM au moins 10 fois supérieure au maximum (1633 Hz)



4 solutions pour Fpwm :

39062 Hz pour 9 bits réels

20 à 25kHz pour 8 bits réels

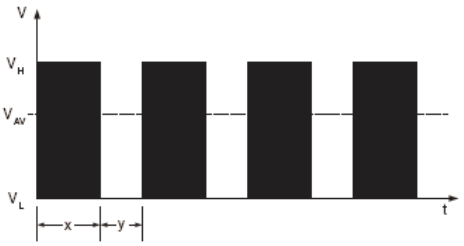
16333 Hz pour 8 ou 7 bits réels

Si on a le temps (CPU libre) on choisira 39062 Hz sur 9 ou 10 bits, sinon, on descendra vers 16333 Hz en 8 bits.

Démarrage PWM

Par défaut, le PWM doit fonctionner avec un rapport cyclique de X% afin que la tension de sortie soit « positionnée » correctement et donc le filtre de sortie ne verra pas d'échelon. Cette valeur est la valeur moyenne des signaux générés, elle doit prendre en compte l'offset installé par la durée de la routine de remise à jour de la consigne.

Cadencement DTMF

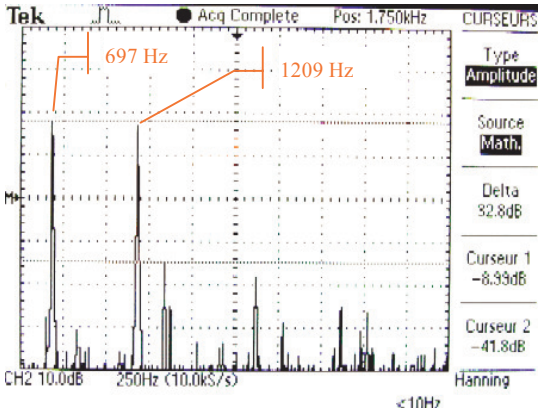


Graphique tiré de AVR314 Atmel

Chaque tonalité est générée pendant Ton, et on doit respecter un silence d'au moins Toff entre 2 émissions.

Implantation et essais

Un algorithme a été implanté sur un PIC18F, avec une table sinus à 256 positions et une fréquence PWM à 39062 kHz. Seule la moitié supérieure de la gamme PWM 10bits est utilisée (0x200 à 0x3FF).
C'est donc un PWM 9 bits qui est implanté ici.



Un fitrage par 2 cellules RC à 3300 Hz (10kΩ+4.7nF puis 100kΩ+470pF) semble suffisant.

La FFT du signal en sortie est propre, on distingue clairement les 2 raies DTMF (697 Hz et 1209 Hz), la première harmonique (1394 Hz) est à –42dB sous le premier fondamental, la suivante (2419 Hz) est encore plus basse, c'est remarquable.

ANNEXE génération d'un sinus par calcul

Si T est a période d'échantillonnage, la transformation en Z d'une sinusoïde est

$$\frac{z^{-1}\sin\omega T}{1-2z^{-1}\cos\omega T+z^{-2}}$$

Si on considère que ce signal est la sortie Y(z) d'un processus dont l'entrée est X(z), alors on a

$$Y(z) = z^{-1}X(z)\sin\omega T + z^{-1}Y(z)2\cos\omega T - z^{-2}Y(z)$$

$$y(n) = \sin\omega T\, x(n-1) + 2\cos\omega T\, y(n-1) - y(n-2)$$

Il est facile d'en déduire les équations simples suivantes :

- $y(0) = 0$
- $y(1) = \sin(\omega T)$
- $y(n) = 2\cos(\omega T)y(n-1) - y(n-2)$ dès que n>1

On peut retrouver cette équation magique en décomposant sin(1+2) = sin(1)cos(2) + sin(2)cos(1)

Le facteur cos(ωT) est constant.
L'algorithme est simple et demande seulement 3 valeurs, mais la précision finale dépend beaucoup de la façon de coder.

Si on utilise des variables 8 bits, autant dire que cette méthode élégante ne donne rien de bon si la fréquence d'échantillonnage est trop forte, ou ωT faible, car l'erreur sur le sinus ou le cosinus est trop forte.

La meilleure précision est obtenue pour F/Fech de l'ordre de 1/6, ωT=pi/3, sin(ωT) =0.866, 2cos(ωT)=1, soit respectivement 110 et 127 pour des valeurs codées en 8 bits signés, les erreurs de calculs seront minimales. Mais ce cas est très marginal et oblige à faire travailler le PWM trop bas.

REFERENCES

- MICROCHIP
  - AN655 : D/A Conversion Using PWM and R-2R Ladders to Generate Sine and DTMF Waveforms ; minable
  - AN616 : Digital Signal Processing with the PIC16C74 ; DTMF PWM à 6500 ech/sec, sans utiliser une lookup table pour générer le sinus…
- ATMEL : AVR314 DTMF Generator, très bonne description